DP

$n + e$

Tsinghua University

2020 年 1 月 28 日

写在前面

- 从状态类型分，并不表示一题只从属于一类.
- 其实一类只是一种状态的表示方法. 可以好几种方法组合成一个状态，来解决问题.
- 时间太紧有些 Point 找不到印象中合适的例题. 我后面会补的.
- 参考了大量资料，感谢 Amber 大神所作的总结!

共性总结

- 本类的状态是基础的基础，大部分的动态规划都要用到它，成为一个维.

共性总结

- 本类的状态是基础的基础，大部分的动态规划都要用到它，成为一个维.
- 一般来说，有两种编号的状态
  1. 状态 $[i]$ 表示前 i 个元素决策组成的一个状态.

共性总结

- 本类的状态是基础的基础，大部分的动态规划都要用到它，成为一个维.
- 一般来说，有两种编号的状态
  1. 状态 $[i]$ 表示前 i 个元素决策组成的一个状态.
  2. 状态 $[i]$ 表示用到了第 i 个元素，和其他在 1 到 $i-1$ 间的元素，决策组成有的一个状态.

Problems

1. LIS
   - 以一元组 $[i]$ 作为状态，表示第 i 个作为序列的最后一个点的时候的最长序列. 于是很容易想到 $O(n^2)$ 的算法. 但本题可合理组织状态，引入一个单调的辅助数组，利用单调性二分查找，优化到 $O(nlogn)$. 1224
   - 一些问题可将数据有序化，转化成本题. 1027
2. LCS
   - 状态 $[i,j]$ 表示第 1 个字符串的第 i 位，与第 2 个字符串的第 j 位匹配，得到的最长的串.
   - 若有多个串要 LCS，则加维，即几个串就几个维.
   - 不是很懂序列自动机的那套理论

共性总结

- 本类问题与下一章的划分问题的决策的分割点无序交集比较大.

Problems

- f[l][r] 表示一段连续区间 $[l, r]$ 上的答案
  1. 序列: 区间 $[l, r]$ 的最后一步处理的是 k 元素 / 分割点是 k / 受 k 控制的
     常用四边形不等式优化
  2. 环: 复制一下粘到后面, 枚举断点
  3. 点集回路: 利用线段不交的性质按顺序 dp

共性总结

- 状态是由坐标维与其他的维组成.
- 划分问题 (2 维或多维的坐标系的划分) 与路径问题的交集占本类问题中大多数.
- 找坐标状态之间的关系: Apio2009-oil

Problems

1. 01 背包：将 Value 作为 dp 的状态
   - 有时候可以用 bitset 优化
   - 装箱问题、币值分割、……
   - 1304
   - 听说排序后有惊喜?
   - 剩余系下跑 spfa：BZOJ2118
2. 博弈论：与 SG 函数结合

共性总结

1. 动态规划的顺序：一般按照后序遍历的顺序，即处理完儿子再处理当前节点，才符合树的子结构的性质.

共性总结

1. 动态规划的顺序：一般按照后序遍历的顺序，即处理完儿子再处理当前节点，才符合树的子结构的性质.
2. 加当前点的选或不选、选几个子树内的节点的常数维，解决后效性问题.

共性总结

1. 动态规划的顺序：一般按照后序遍历的顺序，即处理完儿子再处理当前节点，才符合树的子结构的性质.
2. 加当前点的选或不选、选几个子树内的节点的常数维，解决后效性问题.
3. 要记录的东西：子树内的信息 (、子树外的信息)

共性总结

1. 动态规划的顺序：一般按照后序遍历的顺序，即处理完儿子再处理当前节点，才符合树的子结构的性质.
2. 加当前点的选或不选、选几个子树内的节点的常数维，解决后效性问题.
3. 要记录的东西：子树内的信息 (、子树外的信息)
4. 可以把子树看成背包，信息合并时一颗颗并上去

共性总结

1. 动态规划的顺序：一般按照后序遍历的顺序，即处理完儿子再处理当前节点，才符合树的子结构的性质.
2. 加当前点的选或不选、选几个子树内的节点的常数维，解决后效性问题.
3. 要记录的东西：子树内的信息 (、子树外的信息)
4. 可以把子树看成背包，信息合并时一颗颗并上去
5. 复杂度：树型动态规划复杂度基本上是 O(n)，若有附加维 m，则是 O(nm).

共性总结

1. 动态规划的顺序：一般按照后序遍历的顺序，即处理完儿子再处理当前节点，才符合树的子结构的性质.
2. 加当前点的选或不选、选几个子树内的节点的常数维，解决后效性问题.
3. 要记录的东西：子树内的信息 (、子树外的信息)
4. 可以把子树看成背包，信息合并时一颗颗并上去
5. 复杂度：树型动态规划复杂度基本上是 O(n)，若有附加维 m，则是 O(nm).
6. 无根树的处理：先随便找个根，按有根树处理，从叶子节点处理上去，最后再由根从上往下 dp 一遍. 如求一棵树中每个节点的最远点.

Problems

- 都长一个样

共性总结

- 个人认为类似背包 + 树 + 递推
- 反正 NOIP 又不考 == 后面再说

共性总结

1. 数据特殊性: 给出的数据在某一个或几个维度上一般具有比较小的范围 (可以枚举一类的状态), 但是无法用该范围的数值刻画全部状态.
2. 状态: 选/不选/还没考虑

Problems

- TSP 类: 将一个集合与一个点合并成新的一个集合
  - 子集 dp 转移复杂度为 $O(3^n)$
  - 有可能要记录下该集合内最后一个转移到的点
- 炮兵阵地类: 直接枚举该行的所有状态进行转移
- 插头 DP: 就是把上面的方法改改, 把按“行”转移改成按“行 + 格”转移
  - 连通性? 再见. 这里是 noip 班. 以后再讲

- 这个……应该不用我多说了
- 1802
- 一个 Tip: 考虑 f(n) 与 f(n-1) 和 f(n/2) 之间的关系，缩小问题规模

共性总结

1. 通常都是把题目的输入数据作为状态
2. 考虑状态之间的转移：在原状态添加一个新的元素会有怎样的影响
3. 有些可以直接转换为数学问题，考察数学功底

决策的分割点有序

1. 有序性: 每次决策的点的编号是有序的, 即要按决策的顺序输出分割点的编号的话, 编号是有序的, 满足分割点的编号按升序排列.
2. 方程一般形式:

$$f[n][m] = optimize\{f[k][m-1] + w(k+1, n)\}$$

   f[n][m] 表示从 1 到 n 个点中划分为 m 个部分的最优值, k 为决策的分割点, 即第 m 个部分为 k+1 到 n, 这里 optimize 可以为 max/min.
3. 一般可以使用斜率优化: 1838、2021

决策的分割点无序

1. 无序性: 每次决策的点的编号是无序的, 即要按决策的递归顺序输出分割点的编号的话, 编号是无序的.
2. 方程一般形式:

$$f[i][j] = optimize\{f[i][k-1] + f[k+1][j]\} + w(i,j)$$

f[i][j] 表示从 i 到 j 的范围内选取一个分割点 k 的最优值, 子问题是分割点左边 [i,k-1] 和右边 [k+1,j] 的点的范围的最优值. 这里 optimize 可以为 max/min.
该方程很类似二叉树的性质: 加分二叉树

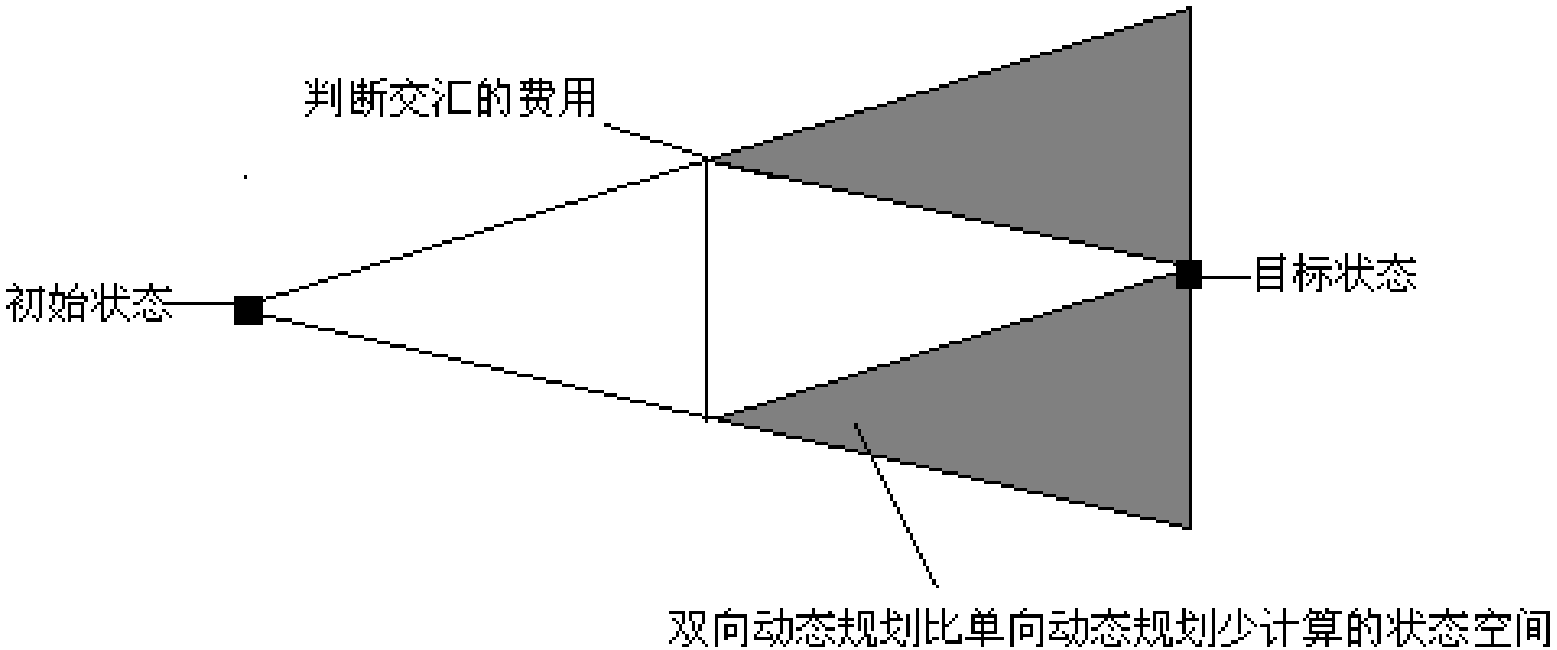
3. 此类问题有些可用四边形不等式优化.

共性总结

1. 行走方向决定阶段性: 有规定源点与终点, 每次行走方向都有一定的规定, 使原点到终点的所有路径形成 DAG.
2. 多源或多汇: 当多源或多汇时, 应该加维, 使得每个源都有一个路径的状态与之对应。
   如有 n 个源的网格类问题, 常常状态是 (x1,y1)(x2,y2)$\cdots$ (xn,yn). 但是源太多的话, 空间上不允许, 可以将问题转成网络流问题.

共性总结

3. 双向动态规划：由于有规定源点与终点，可以双向动态规划，有时由于可用于决策的状态较少，效果就不错了.



共性总结

4. 决策稀疏性: 若对于一个状态, 它的前驱或者后继数很少 (从无环有向图角度就是入度或出度少), 称决策稀疏.
5. 状态稀疏性: 很多状态是没有用的, 如排列的 LCS, 状态为 2 维的 (x,y), 但对于一个 x 只有一个 y, 是有限个. 所以实质上状态数还是线性的.

四边形不等式

- 当函数 $w(i,j)$ 满足 $w(a,c)+w(b,d)\leq w(b,c)+w(a,d)$，且 $a\leq b<c\leq d$ 时，称 $w(i,j)$ 满足四边形不等式.
- 当函数 $w(i,j)$ 满足 $w(i',j')\leq w(i,j)$，且 $i\leq i'<j'\leq j$ 时，称 w 关于区间包含关系单调.
- $s(i,j)=k$ 是指 $f(i,j)$ 这个状态的最优决策

四边形不等式

- 定理一: 如果 w(i,j) 同时满足" 四边形不等式" 和" 决策单调性", 则 f(i,j) 也满足四边形不等式
- 定理二: 当定理一的条件满足时, 让 f(i,j) 取最优值的 k 为 s(i,j), 则 $s(i,j-1) \leq s(i,j) \leq s(i+1,j)$
- 定理三: w 为凸当且仅当 $w(i,j)+w(i+1,j+1) \leq w(i+1,j)+w(i,j+1)$
- 由定理三知, 判断 w 是否为凸即判断 $w(i,j+1)-w(i,j)$ 的值随着 i 的增加是否递减
- 于是求 k 值的时候 s(i,j) 只和 s(i+1,j) 和 s(i,j-1) 有关, 所以可以以 i-j 递增为顺序递推各个状态值最终求得结果, 将 $O(n^3)$ 降为 $O(n^2)$

四边形不等式

- 例题: 1034, 1044, 1160, ...
- 小心发现, 大胆猜想, 不用证明!
- 实践是检验真理的唯一标准

决策单调性

- 做动态规划时常常会见到形如这样的转移方程:

$$f[i] = optimize\{g(j)|L[i] \le j < i\} + w[i]$$

其中 L[i] 随 i 单调不降, 即 $L[1] \le L[2] \le \cdots \le L[n]$, g(j) 表示一个和 j 或 f[j] 有关的函数, w[i] 表示一个和 i 有关的函数

- 有这样一个性质: 如果存在两个数 j,k, 使得 $k \le j$, 而且 $g(k) \le g(j)$ $(opt = max)$ 或 $g(j) \le g(k)(opt = min)$, 则决策 k 是毫无用处的.
- 根据 L[i] 单调的特性, 如果 k 可以作为合法决策, 那么 j 一定可以作为合法决策, 又因为 j 比 k 要优 (注意: 在这个经典模型中, "优" 是绝对的, 与当前正在计算的状态无关), 因此如果把表中的决策按照 j 排序的话, 则 g(j) 必然不降.

决策单调性

- 1635, 1917
- 式子不长那样咋办？

斜率优化

- 一个数列 a[], 可以分成若干组, 一个组 $[L, R]$ 的代价为 $(\sum_{i=L}^{R} a[i])^2 + M$, 求最小代价
- $f[i] = \min\{f[j] + (sum[i] - sum[j])^2 \mid 1 \le j < i\} + M$
- 由于表达式中存在 $sum[i] * sum[j]$ 一项, 因此无法直接用单调队列维护
- 设 $k < j < i$, j 比 k 优

$$f[j] + (sum[i] - sum[j])^2 + M < f[k] + (sum[i] - sum[k])^2 + M$$

- 移项有

$$\frac{sum[j]^2 + f[j] - (sum[k]^2 + f[k])}{2(sum[j] - sum[k])} < sum[i]$$

斜率优化

- 令 $y_j = sum[j]^2 + f[j]$, $x_j = 2 \cdot sum[j]$
- 所以若有

$$K(j,k) = \frac{y_j - y_k}{x_j - x_k} < sum[i]$$

则表示 j 比 k 更优

斜率优化

- 结论: 设 k<j<i, 如果 $K(i,j) < K(j,k)$, 那么 j 点便永远不可能成为最优解, 可以直接将它踢出我们的最优解集.
  1. 如果 $K(i,j) < sum[i]$, 那么就是说 i 点要比 j 点优, 排除 j 点.
  2. 如果 $K(i,j) \geq sum[i]$, 那么 j 点此时是比 i 点要更优, 但是同时由假设有 $K(j,k) > K(i,j) \geq sum[i]$. 这说明还有 k 点会比 j 点更优, 同样排除 j 点.
- 因此在新的单调队列中, 相邻三个点 i,j,k 满足 $K(j,k) \leq K(i,j)$, 这是一个下凸壳
- 剩下的就跟之前讲的没什么两样了

斜率优化

- min 为什么是下凸壳? $f[i] = min\{f[j] + k[i]x[j] | 1 \le j < i\} + b[i]$
- 最小化 $f[j] + k[i]x[j]$. 把式子变形有 $f[j] = -k[i]x[j] + f[i]$
  等价于求直线 $y = -kx + b$ 截点集 $\{(x[i], f[i])\}$ 最小的 b
- max 反过来
- 注意在推导的时候, 不等式两边同乘负数会变号
- 不是 sum[i], 是 h[i], h[i] 不具有单调性: 二分出端点, 剩下一样. 该是什么壳就是什么壳

部分和优化

- 比如预处理表达式中的某些元素，这个大家应该都能看得出来
- 1411 暴力部分，2032
- 难在 dp 的表达式推导而不是优化？

矩阵快速幂

- 如果 dp 转移方程中全是常系数，那么该方程就能很方便的转化为矩阵相乘的形式，求第 n 项的时候就能运用矩阵快速幂在 $log(n)$ 的时间内得到结果，而不用 n 次一次一次递推了.
- 特殊的矩阵乘法也能由 $O(n^3)$ 降至 $O(n^2)$，运用特征方程即可，详见叉姐 (郭晓旭) 的论文.
- 1982, 2172
- 利用自动机的 fail 树来构造转移矩阵，套矩阵快速幂. 2081

数据结构优化

- 树状数组 $f[i] = opt\{f[j] + |g(j)|\}$ 分类讨论求极值
- BZOJ3688 前缀和还带动态修改的
- 线段树 $f[i] = opt\{f[j] + g(j+1, i)\}$, 考虑从 i-1 到 i 之间被修改元素
- 多组询问的括号序列 / 子段和: 把问题分成左右两边, 在合并的时候统计答案即可, 利用分治思想
- 平衡树?